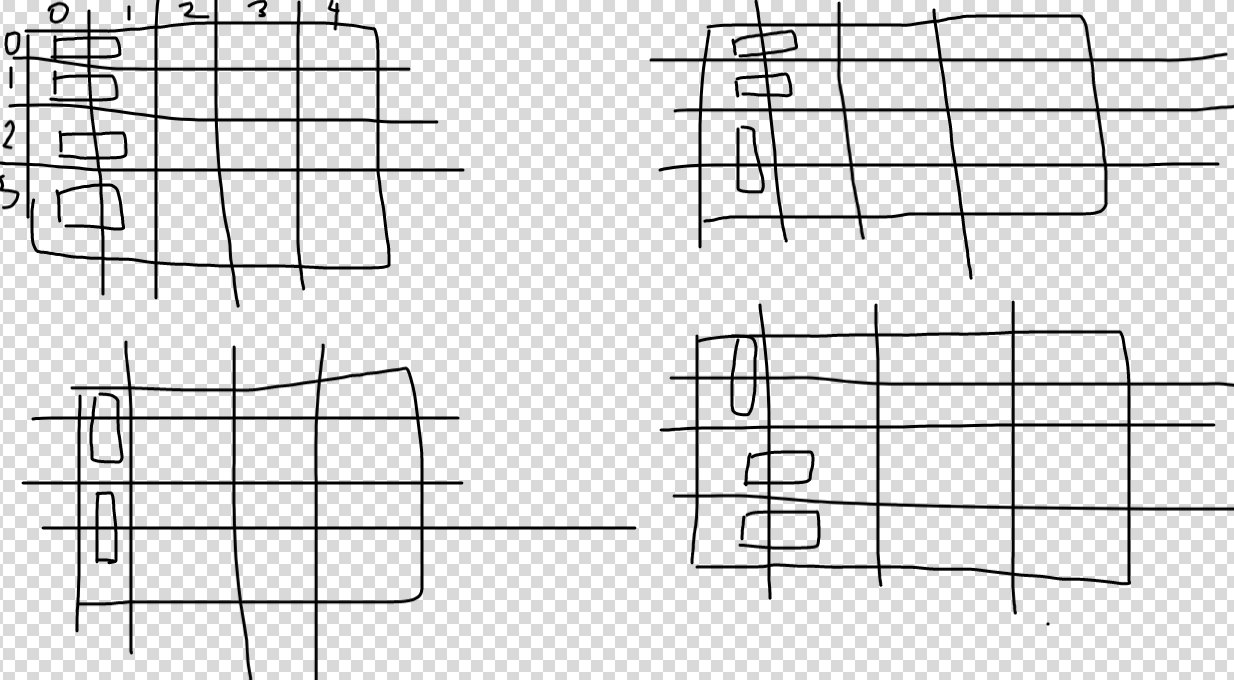
**Problem:**[**https://cses.fi/problemset/task/2181/**](https://cses.fi/problemset/task/2181/)

**Approach:**



-> Idea is simple.

-> (IMP)**We go filling tiles column by column bcoz no of rows are at most 10 and columns are 100 and so it’s easy maintaining the bitmask on ROWS , not columns, (**means which row of a partiqular column is already filled rather than keeping track of which column of a particular row is filled. )

All possibilities for placing tiles in 1st column is given in figure above.

-> Now when we fill the 2nd column, some of the cells in 2nd column are already filled depending on how we filled the 1st column.

-> So we maintain **dp(i,bitmask)** , where ‘i’ is the column no. we are currently at,and bitmask shows which row’s cell in current column is filled and which is empty.

**So in 1st figure, the bitmask for column 1 will be 0000 as all rows are filled, for 2nd figure,**

**It’s 0011,for 3rd figure: 1111 and then 1100 for last one .**

-> All these **4 bitmasks are pre-generated by the 1st column,** and then we call function recursively for 2nd column with these 4 bitmasks only, so that all possibilities are not checked,

And we add the answer returned by all of them.

**-> Base condition : When we reach column no ‘m’ (means out of bound in 0-based indexing), then the bitmask shoulg be all zeros 0000…** means no tile should flow out of last column to out of bounds.

**Code:**[**https://cses.fi/paste/7065431969e7523226a87a/**](https://cses.fi/paste/7065431969e7523226a87a/)

**=> COMPLEXITY ANALYSIS(imp) :**

**-> Lets find number of unique problems we solve :**

-> In worst case take no of rows = 10, now for any column what are the worst case no of ways in which you can place tiles to fill all 10 rows ?

-> f(N) = number of ways to fill N rows of any given column using 0s and 1s given that 0s occur in pairs and lets assume for worst case all cells are empty.

-> At some cell if you fill in a 1 then you get f(N-1) ways and if you fill in a 0 then

immediate position to the right is also a 0 and you get f(N-2) ways.

f(N) = f(N-1) + f(N-2)

f(1) = 1, f(2) = 2

f(10) = 89

-> So how to sum this all up? So the execution of our problem goes column by column.

For 89 unique combinations of placing tiles in current column, there are **at max 89 masks that can be generated for next column by each combination,so transition time to next column is 89 \* 89.**

-> Then for next column although execution reached there 89\*89 times, we solve only 89 unique problems as discussed,and and so again 89 unique problems will go ahead to transition into the 3rd column and as discussed transition time = 89\*89 as 89 masks generated by each and so on…

-> So work done at each column = **no of problems to be solved \* transition time to next column for each = 89\*89.**

So approximate upper bound on time complexity for at max 1000 columns= 89\*89\*1000 ~ 10^7 which pass in 1sec.